PROCESSOR INTERRUPT FILTERING

FIELD OF THE INVENTION

[0001] The invention pertains to the field of processor interrupts and, in particular, to filtering processor interrupts.

BACKGROUND

[0002] In a computer system having processors and peripheral devices, such as a memory or other input/output device, interrupts may be used to request a processor to perform a task with regard to the peripheral device or to execute certain instructions. The processor receives an interrupt request including an interrupt vector where the interrupt vector identifies the instructions or task to be performed by the processor.

SUMMARY

[0003] The following presents a simplified summary of the invention in order to provide a basic understanding of some aspects of the invention. This summary is not an extensive overview of the invention. It is intended to neither identify key or critical elements of the invention nor delineate the scope of the invention. Its sole purpose is to present some general concepts of the invention in a simplified form as a prelude to the more detailed description that is presented later.

[0004] In one embodiment, the invention encompasses a method for processing an interrupt message in a system having a plurality of processors arranged into at least two partitions. The interrupt message is decoded to identify an interrupt source. If the interrupt source is not in an interrupt set, the interrupt is dropped. If the interrupt source is in a local partition, the interrupt is delivered. If the interrupt source is in the interrupt set and not in the local partition, the interrupt is processed in accordance with at least one of a target enable register and a vector enable register.

[0005] In another embodiment, the invention encompasses an apparatus for processing an interrupt in a system having a plurality of processors arranged into at least two partitions. An interrupt set register identifies an interrupt source from which interrupts may be accepted. A partition set register identifies an interrupt source within the same partition as an interrupt target. A target enable register characterizes the enablement of an interrupt target to process interrupts. A vector enable register identifies interrupt vectors enabled for

processing. An interface module receives an interrupt message and includes an interface processor for decoding the received interrupt message to identify at least one of a corresponding interrupt source, interrupt vector, and interrupt target and selectively transmitting the received interrupt message to the interrupt target responsive to at least one of the interrupt source, interrupt vector, interrupt target, interrupt set register, partition set register, target enable register, and vector enable register.

[0006] In yet another embodiment, the invention encompasses a symmetric multiprocessor system comprising a plurality of cells and a routing fabric for communicating a packet from one cell to another cell. A cell includes at least one processor and at least one interface to the routing fabric. The interface includes an interrupt set register identifying a processor from which interrupts may be accepted, a partition set register identifying a processor within the same partition as a target processor, a target enable register characterizing the enablement of a target processor to process interrupts, a vector enable register identifying interrupt vectors enabled for processing, and an interface processor. The interface processor decodes a received interrupt message to identify at least one of a corresponding source processor, interrupt vector, and target processor and selectively transmits the received interrupt message to the target processor responsive to at least one of the source processor, interrupt vector, target processor, interrupt set register, partition set register, target enable register, and vector enable register.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] For the purpose of illustrating the invention, there is shown in the drawings a form that is presently preferred; it being understood, however, that this invention is not limited to the precise arrangements and instrumentalities shown.

[0008] Figure 1 is a flow chart of a method of processing an interrupt according to an embodiment of the invention;

[0009] Figure 2 is a block diagram of a system for processing an interrupt according to an embodiment of the invention;

[0010] Figure 3 is a block diagram of a multiprocessor system according to an embodiment of the invention; and

[0011] Figure 4 is a block diagram of an interface according to an embodiment of the invention.

DETAILED DESCRIPTION

[0012] Referring to the drawings, in which like reference numerals indicate like elements, there is shown in Figure 1 a flow chart 100 illustrating a method of processing an interrupt message or transaction described with reference to a system 200 shown in Figure 2.

[0013] The system 200 includes a plurality of nodes 202 coupled to each other via a communication medium 204. Each node 202 includes at least one processor and an interface to the communication medium 204. A method of processing interrupts is executed by the interface in an embodiment of the invention.

[0014] One or more nodes 202 are grouped to form partitions illustrated by the dashed lines 206. In an embodiment, partitions 206 are used to segregate a processor in one node from a processor in another node that may be running under different operating systems (e.g., Windows, Unix) or different applications.

[0015] A processor in one node 202 ("interrupt source") may generate an interrupt for a processor in another node 202 ("interrupt target"). In an embodiment, a node 202 in the system may send a message (interrupt message or transaction) to a processor on another node 202 via an interrupt. In an embodiment, the interrupt message informs or instructs another node 202 to perform a particular task. Such a task may, for example, comprise instructing a node to add another node to its partition. A resource such as memory may be shared by allowing access to the resource by another node 202 via an interrupt. In an embodiment, the nodes 202 may generate an interrupt message to access the memory of another node 202 on a read-only basis and bi-directional communication (or interrupts) between nodes are implemented by generating and processing interrupts in both direction between nodes.

[0016] When one node 202 generates in interrupt, it is transmitted over the communications medium 204 to another node 202. The term "interrupt source" is used herein to identify the processor generating the interrupt or the node comprising that processor. The term "interrupt target" is used herein to identify the processor to which the interrupt is targeted or the node comprising that processor.

[0017] An interrupt message from an interrupt source is received in step 102 and is decoded to determine an interrupt source identifier. An interrupt set identifies interrupt sources (processors or cells) from which interrupts may be processed. An exemplary interrupt set is illustrated in Figure 2 by the line 210 encompassing several partitions 206. An interrupt received from an interrupt source is processed if the interrupt source is identified in the interrupt set and is discarded or dropped if the interrupt source is not identified in the interrupt set. The interrupt source is determined to be in or out of the interrupt set in step

104. If the interrupt source is determined to not be in the interrupt set, the interrupt is dropped (e.g., discarded and not processed) in step 106.

[0018] In an embodiment, the interrupt set identifies processors within the system 200 from which interrupts may be accepted and the interrupt source identifier identifies the processor which originated a corresponding interrupt. In another embodiment, the interrupt set identifies nodes 202 from which interrupts may be accepted and the interrupt source identifies the node which originated a corresponding interrupt. For example, a packet including an interrupt request may include a source address that identifies the processor or the node that includes the processor that originated the request.

[0019] The system 200 determines in step 108 whether the interrupt source is in a local partition where the interrupt source is "local" if it is in the same partition that includes the interrupt target. If the interrupt source is within the interrupt set and within the local partition, the interrupt is delivered to the interrupt target in step 110. If the interrupt source is in the interrupt set but not in the local partition, the interrupt is processed in accordance with at least one of a target enable register and a vector enable register.

[0020] A vector enable register characterizes the enablement of interrupt vectors. In an embodiment, an interrupt message includes an n-bit interrupt vector and the vector enable register includes 2^n bits, one for each combination of the n bits in the interrupt vector. Each bit of the vector enable register defines whether a particular combination of the n bits in an interrupt vector is enabled or disabled. In an embodiment, a bit value of "1" in the vector enable register enables a corresponding interrupt vector and a bit value of "0" disables the corresponding interrupt vector.

[0021] A target enable register characterizes the enablement of a interrupt target (processor or node) to process an interrupt. In an embodiment, the target enable register includes one bit corresponding to each of the interrupt targets (processor or node) in the system 200 and a bit value of "1" identifies a corresponding interrupt target as being enabled to process interrupts and a bit value of "0" identifies a corresponding interrupt target as being disabled.

[0022] In the embodiment illustrated in Figure 1, the processing of an interrupt is described in accordance with a target enable register and a vector enable register. The interrupt is decoded to identify an interrupt target in step 122. The system 200 determines whether the interrupt target is enabled to accept interrupts based on the target enable register in step 112. If the interrupt target is not enabled to process interrupts, the error is logged in step 114 and the interrupt is dropped in step 106. In an embodiment, an error is logged by incrementing a counter corresponding to an action taken. In another embodiment, an error is

logged by storing a record identifying parameters relating to the interrupt request such as the interrupt source, the interrupt vector, and the interrupt target.

[0023] If the interrupt target is enabled to process interrupts, the interrupt message is decoded to identify an interrupt vector and in step 116 the system 200 determines whether the interrupt vector is enabled based on the vector enable register. If the interrupt vector is enabled, the interrupt is delivered to the interrupt target in step 110. If the interrupt vector is not enabled, the interrupt vector is replaced with an error vector in step 118, the non-enablement is logged in step 120, and the error vector is delivered to the interrupt target in step 110. The log creates a record that an incorrect vector was received.

[0024] Although the steps in the flow chart 100 shown in Figure 1 are illustrated has being in a particular order, the invention encompasses embodiments of the method where steps of the method are implemented out of the listed order. For example, in an embodiment where the local partition is a subset of the interrupt set, step 108 may be performed before step 104. As another example, the step 116 may be performed before step 112.

[0025] The method of processing interrupts described above with reference to Figure 1 provides a level of security in inter-node communication by allowing restrictions to be enforced based on target enable and vector enable register values. In an embodiment, interrupts initiated by input/output adaptors are excluded from the interrupt set so they are not processed.

[0026] An interrupt source from the interrupt set may flood a processor, node, or local partition with excessive interrupts. In an embodiment, an interrupt source is removed from the interrupt set, to prevent a flood of interrupts for example, in response to the number of interrupts, the number of interrupts of having a particular characteristic, or the number of logged errors resulting from interrupt messages received from the interrupt source. Exemplary characteristics of interrupts include interrupts that are erroneous, unexpected, from a particular interrupt source, or from outside the local partition.

[0027] A multiprocessor system according to an embodiment of the invention is shown in Figure 3. A node (or cell) 302 comprises one or more processors 304 (a node processor) that are coupled to another node 302 via a routing fabric 306 comprising at least one router 308. One or more nodes 302 are grouped into partitions 312. The processors 304 are coupled to the routing fabric 306 via one or more interface blocks 310. In an embodiment of the invention, the system 300 is a symmetric multiprocessor system where the routers 308 are crossbars.

[0028] In an embodiment, the interface block 310 comprises an interface processor for processing an interrupt message. A block diagram of an interface module 310 according

to an embodiment of the invention is shown in Figure 4. The interface module 310 stores an interrupt set register 402, a partition set register 404, a target enable register 406, and a vector enable register 408 and processes a received interrupt message in accordance with the method of processing an interrupt message described above. In an embodiment, the interface block 310 performs the method illustrated in Figure 1.

[0029] The interface module 310 receives an interrupt message (transaction) on an input queue 410 which segregates the received interrupt message into a header portion and a data portion. In an embodiment, the header portion comprises an identifier of the interrupt source and the data portion comprises an interrupt vector. In the embodiment shown in Figure 4, the interrupt processor comprises a header processor 412 and a data processor 414. The header portion is transmitted to and processed by a header processor 412 and the data portion is transmitted to and processed by a data processor 414.

[0030] The header processor 412 and data processor 414 process the interrupt message in response to control signals 416, 418 received from a control block 420. The header processor 412 decodes an interrupt source identifier from the header portion and provides it to the control block 420. The data processor 414 decodes an interrupt vector from the data portion and provides it to the control block 420. The control block 420 processes a received interrupt message and determines when and if to provide the interrupt to a node processor. When the control block 420 determines to provide an interrupt to a node processor, it provides control signals 416, 418 to the header and data processors 412, 414 indicating to output the header and data to the header and data queues 422, 424, respectively. In an embodiment, the header and data processors process interrupt messages in parallel.

[0031] The foregoing describes the invention in terms of embodiments foreseen by the inventors for which an enabling description was available, although insubstantial modifications of the invention, not presently foreseen may nonetheless represent equivalents thereto.